

#3

Docket No.: 50023-165

PATENT

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Application of

Yuko KUBOOKA, et al.

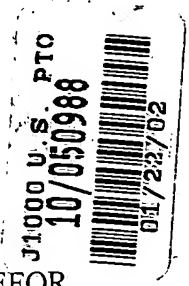
Serial No.:

Group Art Unit:

Filed: January 22, 2002

Examiner:

For: DIGITAL DEVICE, TASK MANAGEMENT METHOD AND PROGRAM THEREFOR



**CLAIM OF PRIORITY AND  
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT**

Commissioner for Patents  
Washington, DC 20231

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicants hereby claim the priority of:

**Japanese Patent Application No. 2001-016601, filed January 25, 2001**

cited in the Declaration of the present application. A Certified copy is submitted herewith.

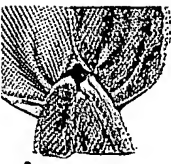
Respectfully submitted,

MCDERMOTT, WILL & EMERY

A handwritten signature in black ink, appearing to read "S. A. Becker".

Stephen A. Becker  
Registration No. 26,527

600 13<sup>th</sup> Street, N.W.  
Washington, DC 20005-3096  
(202) 756-8000 SAB:prp  
**Date: January 22, 2002**  
Facsimile: (202) 756-8087



日 本 国 特 許 庁  
JAPAN PATENT OFFICE

Kubooka et al.  
January 22, 2002  
McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日  
Date of Application:

2001年 1月25日

出 願 番 号  
Application Number:

特願2001-016601

出 願 人  
Applicant(s):

松下電器産業株式会社

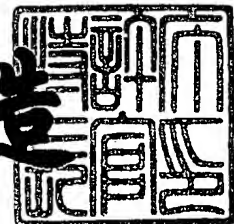
J1000 U.S. PTO  
10/050988  
01/22/02

CERTIFIED COPY OF  
PRIORITY DOCUMENT

2001年11月26日

特許庁長官  
Commissioner,  
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3102964

【書類名】 特許願

【整理番号】 2022530017

【提出日】 平成13年 1月25日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 9/46

【発明者】

    【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

    【氏名】 久保岡 祐子

【発明者】

    【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

    【氏名】 土井 繁則

【特許出願人】

    【識別番号】 000005821

    【氏名又は名称】 松下電器産業株式会社

【代理人】

    【識別番号】 100083172

    【弁理士】

    【氏名又は名称】 福井 豊明

【手数料の表示】

    【予納台帳番号】 009483

    【納付金額】 21,000円

【提出物件の目録】

    【物件名】 明細書 1

    【物件名】 図面 1

    【物件名】 要約書 1

    【包括委任状番号】 9713946

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 デジタル機器、タスク管理方法及びそのプログラム

【特許請求の範囲】

【請求項 1】 一実行単位として管理されるタスクを、プログラム実行手段にて複数並列に実行するオペレーティングシステムを備えたデジタル機器において

上記タスクを構成する関数の属性を記憶するタスク属性情報記憶手段と、

実行されている関数の属性に関する情報をタスク属性認識手段に送信すると共に、当該関数の属するタスクに対する強制終了の実行をタスク実行判断手段に問い合わせる上記プログラム実行手段と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されている関数の属性に基づいて上記強制終了の実行を判定するタスク実行判断手段と、

上記プログラム実行手段から送信された上記関数の属性に関する情報に基づいて、実行されている関数の属性を上記タスク属性情報記憶手段に格納すると共に、上記タスク実行手段からの実行されている関数の属性の問い合わせに対して、上記タスク属性情報記憶手段に格納した上記関数の属性を返信するタスク属性認識手段とを具備することを特徴とするデジタル機器。

【請求項 2】 上記関数の属性に関する情報が、実行されている関数の名称であると共に、

上記タスク属性認識手段は、上記関数のコードと、上記名称に基づいて上記実行されている関数の属性を判定する請求項 1 に記載のデジタル機器。

【請求項 3】 上記一実行単位として管理されるタスクは、プロセスである請求項 1 に記載のデジタル機器。

【請求項 4】 上記一実行単位として管理されるタスクは、スレッドである請求項 1 に記載のデジタル機器。

【請求項 5】 一実行単位として管理されるタスクを、複数並列に実行するオペレーティングシステムにおけるタスク管理方法において、

上記タスクを構成する関数であって、現在実行されている関数の属性を記憶し

上記実行されている関数の属するタスクに対する強制終了の実行時に、上記記憶した関数の属性に基づいて、上記実行されている関数の強制終了の可否を判定することを特徴とするタスク管理方法。

【請求項 6】 上記関数の属性が、実行されている関数の名称を用いて決定される請求項 5 に記載のタスク管理方法。

【請求項 7】 上記一実行単位として管理されるタスクは、プロセスである請求項 5 に記載のタスク管理方法。

【請求項 8】 上記一実行単位として管理されるタスクは、スレッドである請求項 5 に記載のタスク管理方法。

【請求項 9】 一実行単位として管理されるタスクを、プログラム実行機能にて複数並列に実行するオペレーティングシステムをコンピュータに実現させるプログラムにおいて、

実行されている関数の属性に関する情報をタスク属性認識機能に送信すると共に、当該関数の属するタスクに対する強制終了の実行をタスク実行判断機能に問い合わせる上記プログラム実行機能と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されている関数の属性に基づいて上記強制終了の実行を判定するタスク実行判断機能と、

上記プログラム実行機能から送信された上記関数の属性に関する情報に基づいて、実行されている関数の属性を記憶手段に格納すると共に、上記タスク実行機能からの実行されている関数の属性の問い合わせに対して、上記記憶手段に格納した上記関数の属性を返信するタスク属性認識機能とをコンピュータに実現させるためのプログラム。

【請求項 10】 一実行単位として管理されるタスクを、プログラム実行機能にて複数並列に実行するオペレーティングシステムをコンピュータに実現させるプログラムにおいて、

実行されている関数の属性に関する情報をタスク属性認識機能に送信すると共に、当該関数の属するタスクに対する強制終了の実行をタスク実行判断機能に問い合わせる上記プログラム実行機能と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されている関数の属

性に基づいて上記強制終了の実行を判定するタスク実行判断機能と、

上記プログラム実行機能から送信された上記関数の属性に関する情報に基づいて、実行されている関数の属性を記憶手段に格納すると共に、上記タスク実行機能からの実行されている関数の属性の問い合わせに対して、上記記憶手段に格納した上記関数の属性を返信するタスク属性認識機能とを実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項 11】 一実行単位として管理されるタスクを、プログラム実行手段にて複数並列に実行するオペレーティングシステムを備えたデジタル機器において、

上記タスクの属性を記憶するタスク属性情報記憶手段と、

上記タスクの属性に関する情報をタスク属性認識手段に送信すると共に、当該タスクに対する強制終了の実行をタスク属性判断手段問い合わせる上記プログラム実行手段と、

上記強制終了の実行の問い合わせに対し、少なくともタスクの属性に基づいて上記強制終了の実行を判定するタスク実行判断手段と、

上記プログラム実行手段から送信された上記タスクの属性に関する情報に基づいて、当該タスクの属性を上記タスク属性情報記憶手段に格納すると共に、上記タスク実行手段からのタスクの属性の問い合わせに対して、上記タスク属性情報記憶手段に格納した上記タスクの属性を返信するタスク属性認識手段とを具備することを特徴とするデジタル機器。

【請求項 12】 上記一実行単位として管理されるタスクは、プロセスである請求項 11 に記載のデジタル機器。

【請求項 13】 上記一実行単位として管理されるタスクは、スレッドである請求項 11 に記載のデジタル機器。

【請求項 14】 一実行単位として管理されるタスクを、複数並列に実行するオペレーティングシステムにおけるタスク管理方法において、

実行されている上記タスクの属性を記憶し、

上記タスクに対する強制終了の実行時に、上記記憶したタスクの属性に基づいて、上記実行されているタスクの強制終了の可否を判定することを特徴とするタ

スク管理方法。

【請求項 1 5】 上記一実行単位として管理されるタスクは、プロセスである請求項 1 4 に記載のタスク管理方法。

【請求項 1 6】 上記一実行単位として管理されるタスクは、スレッドである請求項 1 4 に記載のタスク管理方法。

【請求項 1 7】 一実行単位として管理されるタスクを、プログラム実行機能にて複数並列に実行するオペレーティングシステムをコンピュータに実現させるプログラムにおいて、

実行されているタスクの属性に関する情報をタスク属性認識機能に送信すると共に、当該タスクに対する強制終了の実行をタスク実行判断機能に問い合わせる上記プログラム実行機能と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されているタスクの属性に基づいて上記強制終了の実行を判定するタスク実行判断機能と、

上記プログラム実行機能から送信された上記タスクの属性に関する情報に基づいて、実行されているタスクの属性を記憶手段に格納すると共に、上記タスク実行機能からの実行されているタスクの属性の問い合わせに対して、上記記憶手段に格納した上記タスクの属性を返信するタスク属性認識機能とをコンピュータに実現させるためのプログラム。

【請求項 1 8】 一実行単位として管理されるタスクを、プログラム実行機能にて複数並列に実行するオペレーティングシステムをコンピュータに実現させるプログラムにおいて、

実行されているタスクの属性に関する情報をタスク属性認識機能に送信すると共に、当該タスクに対する強制終了の実行をタスク実行判断機能に問い合わせる上記プログラム実行機能と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されているタスクの属性に基づいて上記強制終了の実行を判定するタスク実行判断機能と、

上記プログラム実行機能から送信された上記タスクの属性に関する情報に基づいて、実行されているタスクの属性を記憶手段に格納すると共に、上記タスク実行機能からの実行されているタスクの属性の問い合わせに対して、上記記憶手段

に格納した上記タスクの属性を返信するタスク属性認識機能とを実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、デジタル機器、タスク管理方法及びそのプログラムに関し、詳しくは、マルチタスクオペレーションシステムにてタスクの強制終了を管理するデジタル機器、タスク管理方法及びそのプログラムに関する。

【0002】

【従来の技術】

従来では、例えばSTB（set-top box：ケーブルTVのコントロールボックス等、家庭用テレビに接続して追加機能を提供するデバイスの一般名称）や携帯電話等、主にソフトウェアを用いて様々な機能を実現する製品の場合、当該ソフトウェアは上記製品に、OS（オペレーションシステム）及び各機能を実現するプログラムを予め格納した、いわゆる組込OSという形で提供される。上記組込OSを備えた製品を提供するメーカーは、当該組込OSの各機能の動作を十分に検証し、様々な動作状態におけるデバッグを行うことができるため、安定動作する信頼性の高い製品を提供することが可能である。

【0003】

又、近年では、製品によってはライフタイムが短く、又新機能の追加・変更サイクルも短くなっている。このため、当該機能追加・変更に対応するべく、上記組込OSに替えて汎用OSを用いるケースが増えている。さらにネットワークの発達（多様化）やJavaに代表されるネットワークに適した多機能なインタプリタ言語の普及により、機能追加・変更に伴うプログラムのダウンロードが容易になってきている点も、上記組込OSの汎用OS化を促進する要因となっている。

【0004】

以下に上記汎用OSを用いた際の一般的なタスクの実行形態を説明する。尚、上述したような汎用OSは通常マルチタスクとして動作するため、該マルチタス



クを含めた説明を行う。

【0005】

マルチタスクOSでは、図5(a)に示すように、OS501上で例えば複数のプロセス1、2(502、503)が実行される。又、上記プロセス1(502)の管理下にて、さらに例えばスレッド1(504)、スレッド2(505)、上記プロセス2(503)の管理下では例えばスレッド4(506)という様に、複数のスレッドが実行される。尚、例えば上記プロセス1(502)は、実行時に上記OS501よりメモリ空間やI/O(Input/Output)空間等のリソース(資源)が割り当てられ、当該リソースは上記スレッド1(504)及びスレッド2(505)にて共有される。

【0006】

ここで、上記プロセスとは、プログラムの一実行単位を指す。即ち、上記リソース等を独立して所有すると共に、切り替え時(マルチタスクOSによるプロセスの切り替え時)においてCPUレジスタの内容をすべて保存し、次に制御を切り替えるプロセスのためのレジスタ値をロードすることにより各実行単位の独立を保障している単位である。又、上記スレッドもプログラムの一実行単位であり、同一プロセス内でのマルチタスク処理を可能にしたものである。尚、上記プロセスやスレッドという単位は、OSにより扱いが異なるが本質はプログラムの一実行単位であり、関数の集合といえることができる。このため以降、適宜プロセスとスレッドを使い分けて説明するが、双方ともタスクであり、プログラムの一実行単位である。さらに、アプリケーションも広義のタスクである。

【0007】

次に、上述したタスク(プロセス及びスレッド)の構造について簡単に説明する。上記タスクは一般的に、関数の集合により構成されており、例えば所定の関数が別の関数に引数を渡し、当該別の関数が計算した戻り値を用いて次の処理を実行するといった処理が行われる。例えば図6を用いて説明すると、タスク(スレッド1)601は、ユーザが開発したアプリケーション関数a(602)をメインルーチン(メイン関数)とし、当該メインルーチンの処理の過程で、アプリケーション関数b(603)を呼び出している。当該アプリケーション関数b(

603) は、さらに必要に応じてシステムライブラリ関数として提供されるシステムライブラリ関数 a (604) を読み出し、さらに当該システムライブラリ関数 a (604) はアプリケーション関数 c (605) を呼び出している。上記処理の過程で呼び出された関数は、それぞれ呼び出し時に与えられた引数を用いて戻り値を返し、それに応じて呼び出した関数は続く処理を実行する。

#### 【0008】

以上のように、タスクを構成する複数の関数がそれぞれ処理されることにより、所定のタスク（メインルーチン）の処理が完了する。ここに、上記アプリケーション関数とは、例えば当該タスクをプログラミングした者が必要に応じて作成した関数であり、システムライブラリ関数とは、予め提供されている基本的な関数等である。尚、上記アプリケーション関数及びシステムライブラリ関数共、例えば数十から数千のステップ（命令）にて構成されている。

#### 【0009】

さて、上述した各スレッド1 (504) 及びスレッド2 (505) は、図5 (b) に示すように、上記OS 501にて順次割り当てられた処理時間において実行される。即ち、OS 501は、タイマー処理にてスレッド1とスレッド2（ここではスレッド3の処理は考慮しないものとする）それぞれに処理時間507～510を与える。ここでは、例えば、まず処理時間507がOS 501により割り当てられた時にスレッド1が実行され、続いて割り当てられる処理時間508には、スレッド2が実行される。このように、OSが処理時間を複数のスレッド（タスク）に順次与えることにより、複数の処理を並列に実行することが可能となり、即ちマルチタスクOSを実現可能となる。尚、上記スレッド（タスク）の切り替えをディスパッチという。

#### 【0010】

以上に説明した汎用OSを用い、さらにネットワークの発達（多様化）や多機能なインタプリタ言語の普及等により、様々な製品への新たな機能追加等を柔軟に行うことが可能になっている。

#### 【0011】

尚、新たに機能追加等によるシステムの変更後には、当該新機能を組み合わせ

た様々な使用形態が考えられる。このような状態では、通常、リソースが大量に消費されてしまい、後に実行されるプログラムにリソースを割り当てることができず、このため新たなプログラムを起動できなくなる。このような状態を回避するために、従来では、例えば他のタスク（プロセスやスレッド）の処理を終了させる命令、即ち強制終了命令を発行することが可能である。当該強制終了命令により、異なるタスクのリソースを開放し、新たなプログラムを起動することが可能となる。上記リソースの不足は、ある程度機能が限定されている製品（例えば上記STB、携帯電話等）においては、リソースを十分に備えていないことが多く、汎用のパーソナルコンピュータとは異なり起こりやすいといえる。

## 【0012】

但し、むやみに他のタスクの処理を強制終了可能とすると、当該他のタスクがデータの書き込み等の重要な処理を行いつつ最中に終了してしまうといったことが考えられる。このため、特開平10-69392に記載の技術では、所定の処理区間に対して「強制終了禁止区間（アボート禁止区間）」を設定可能とすることにより当該処理区間における強制終了の禁止を可能としている。

## 【0013】

以下図7を用いて、タスクの処理を終了させる命令の発行の仕組み及び、強制終了禁止区間の設定について説明する。

## 【0014】

図7において、スレッド1（601）、及びスレッド2（701）がOS上で独立したタスクとして起動しているものとする。尚、上記スレッド1（601）及びスレッド2（701）は、上述したマルチタスク処理にて、随時切り替えられながら処理を実行しており、例えばスレッド1（601）に割り当てられた処理時間が終了すると、次にスレッド2（701）が動作する。

## 【0015】

ここで、スレッド1（601）におけるシステムライブラリ関数a（604）の実行時702に処理が上記スレッド2（701）に切り替えられた場合、スレッド2（701）は例えばアプリケーション関数d（703）から処理を実行する。

## 【0016】

続いてスレッド2 (701) におけるアプリケーション関数 e (704) の処理の所定のポイント705にて、スレッド2 (701) が、スレッド1 (601) に強制終了命令を発行し、アプリケーション関数 d (703) の所定のポイント706にてスレッド1 (601) に処理が切り替わったと仮定する。この場合には、例えばスレッド1 (601) におけるシステムライブラリ関数 a (604) 実行時に強制終了命令を受け付けることになり、当該システムライブラリ関数 a (604) 実行中 (上記実行時702直後) にスレッド1 (601) が例えばOSにより強制終了されることになる。尚、上記各関数 (602~605) による処理が強制終了されると、全体の処理に大きな影響を与える場合等には以下のような強制終了禁止区間が設けられる。

## 【0017】

即ち、スレッド1 (601) におけるアプリケーション関数 a (602) の所定のポイント707にて、当該アプリケーション関数 a (602) は強制終了禁止命令708を発行する。当該強制終了禁止命令は、例えばOSにおける所定の処理にて認識され、記憶される。次に、上記同様、スレッド2 (701) から所定のポイント705で強制終了命令が発行され、処理がスレッド1 (601) に切り替えられると、例えばOSが、上記強制終了禁止命令が発行されているか否かを判定し、ここでは強制終了禁止命令が発行されているため強制終了せずに通常の処理を継続する。

## 【0018】

続いてアプリケーション関数 a (602) が所定のポイント709にて、強制終了許可命令710を発行すると、当該強制終了許可命令は例えばOSにおける所定の処理にて認識され、強制終了禁止命令708発行時から強制終了許可命令710の発行時までの区間、即ち、強制終了禁止区間が終了する。

## 【0019】

上記強制終了禁止区間が終了すると、例えばOSは、上記所定のポイント705にてスレッド2 (701) により発行された強制終了命令を反映し、アプリケーション関数 a (602) における所定のポイント709にてスレッド1 (60

1) の強制終了を行う。

【0020】

以上により、タスク間（スレッド間）にて強制命令を発行することで、例えば他のタスクが使用しているリソースを解放させることが可能となる。又、システムに影響を与えるような重要な処理（例えば共有リソースに対する操作）を行っている場合には、他のタスクによる強制終了を禁止することで、当該重要な処理の正常終了を可能としている。

【0021】

【発明が解決しようとする課題】

しかしながら、上記従来技術を用いた場合には以下のような問題が生じる。即ち、上述したように、ネットワークを用いて例えば新機能を備えたプログラム（ソフトウェア）を汎用OSにダウンロードして実行した場合には、組込OSと異なり、各機器により異なる複数のプログラムが実行されている等、ユーザー環境が多岐に渡るため正常動作しないことが考えられる。このような場合には、上記プログラムを強制終了させる必要があるが、上述した強制終了禁止区間が設定されてしまうと、強制終了することができないといった問題が起こる。

【0022】

又、特にインターネットを介して第三者が作成したプログラムをダウンロードする場合には、当該プログラムは必ずしも善意の基に作成されたものとは限らないため、当該プログラムを実行するには危険が伴う。

【0023】

即ち、例えば図7に示したスレッド1（601）が悪意を持ったプログラムであった場合、具体的には、例えばアプリケーション関数b（603）内にて無限ループし、かつ同様のアプリケーション関数b（603）を持ったスレッドを生成するプログラムであった場合が考えられる。このような場合には、上記悪意を持ったプログラムを実行してしまうと、スレッドが増殖し、さらに当該スレッドに対する強制終了命令も、上述した強制終了禁止区間の設定により無効になってしまう。こうなると、上記プログラムを実行した端末（製品）は、OSの制御が効かなくなり、最終的には電源のオフ等が必要になってしまう。

## 【 0 0 2 4 】

このような現象は、汎用OSを用いて特定の機能を提供する製品に限らず、一般的な汎用OSであるWindowsやUNIXでも考えられることである。

## 【 0 0 2 5 】

そこで本発明は、上記従来の事情に基づいて提案されたものであって、システムに障害をもたらす可能性のあるタスク、例えば共有リソースの操作中には、当該タスクを強制終了させないタスク管理方法を提供することを目的とする。

## 【 0 0 2 6 】

## 【課題を解決するための手段】

本発明は、上記目的を達成するために以下の手段を採用している。すなわち、本発明は、一実行単位として管理されるタスクを、プログラム実行手段にて複数並列に実行するオペレーティングシステムを備えたデジタル機器を前提としている。ここで、プログラム実行手段は、実行されている関数の属性に関する情報をタスク属性認識手段に送信すると共に、当該関数の属するタスクに対する強制終了の実行をタスク実行判断手段に問い合わせる。又、タスク事項判断手段は、上記強制終了の実行の問い合わせに対し、少なくとも実行されている関数の属性に基づいて上記強制終了の実行を判定する。さらに、タスク属性認識手段は、プログラム実行手段から送信された上記関数の属性に関する情報に基づいて、実行されている関数の属性をタスク属性情報記憶手段に格納すると共に、上記タスク実行手段からの実行されている関数の属性の問い合わせに対して、上記タスク属性情報記憶手段に格納した上記関数の属性を返信する。

## 【 0 0 2 7 】

以上の構成では、関数の呼び出し時、及び処理が戻った時に、タスクがタスク属性認識手段に以後実行される関数の種類を通知することで、タスク属性情報記憶手段には、現在実行されているタスクの種類が常時格納され、このため、タスク属性情報記憶手段に格納されている、現在のタスクにて実行されている関数の属性情報を参照・判定可能となる。従って、強制終了禁止区間であっても、所定の関数を実行している場合には、強制終了可能とし、一方でシステムに障害をもたらす可能性のある別の関数の実行時には、当該タスクを強制終了させないとい

った強制終了についての柔軟な制御が可能である。

【 0 0 2 8 】

又、上記関数の属性に関する情報を実行されている関数の名称とし、タスク属性認識手段が、上記関数のコードと、上記名称に基づいて上記実行されている関数の属性を判定する構成もある。

【 0 0 2 9 】

この構成では、言語によっては、当該言語の仕様により関数の属性を判断し、直接上記タスク属性認識手段に通知することが困難な場合があが、このような場合でも実行している関数の属性を確認することができる。

【 0 0 3 0 】

尚、上記一実行単位であるタスクは、プロセスである場合や、スレッドである場合がある。

【 0 0 3 1 】

さらに、タスク属性情報記憶手段が、タスクを構成する関数の属性に代えてタスク自体の属性を格納することにより、タスクの属性に基づいた強制終了の制御を可能とすることができる。

【 0 0 3 2 】

尚、プログラム及びプログラム記録媒体は、コンピュータにて上記各手段の機能を実現する構成である。

【 0 0 3 3 】

【発明の実施の形態】

以下、添付図面を参照して、本発明の実施の形態につき説明し、本発明の理解に供する。尚、以下の実施の形態は、本発明を具体化した一例であって、本発明の技術的範囲を限定する性格のものではない。

【 0 0 3 4 】

〔実施の形態 1〕

本発明の実施の形態 1 におけるデジタル機器 1 0 0 について図 1、図 2、図 4 を参照し説明する。ここに、上記デジタル機器 1 0 0 とは、例えば S T B、携帯電話、家電、パーソナルコンピュータ等であって、外部からのアプリケーション

(プログラム)を付加(追加)等することにより、当該デジタル機器への所定機能の追加・変更・修正が可能である機器を示す。

【0035】

まず、上記デジタル機器100は、例えばインターネット等のネットワーク101を介して所定のアプリケーション(プログラム:102)を取得し、記憶手段103に記憶する。ここに、当該アプリケーション102は、例えば上記デジタル機器100に新たな機能を追加するためのインタプリタ言語にて記述されたプログラムであり、実行時には、図2に示すタスク201を生成して処理を実行するものとする。尚、上記アプリケーション102は、必ずしもネットワーク101を介する必要は無く、メモリーカード等の記憶媒体を介して得るものであってもよい。

【0036】

次に、上記アプリケーション102は、実行時には例えばローダー等により、RAM(Random Access Memory)等の一時記憶手段105に読み出される(図4:S401)。尚、上記アプリケーション102は、図2のタスク201に示すように、アプリケーション関数a(203)、アプリケーション関数b(204)、アプリケーション関数c(206)より構成される。さらに、上記アプリケーション102は、実行時にシステムライブラリ関数である、システムライブラリ関数a(205)を呼び出すため、例えば上記ローダーにて読み出される際に、同時に上記記憶手段103に格納されているシステムライブラリ104から、システムライブラリ関数107が上記一時記憶手段105に読み出される。

【0037】

続いて、上記アプリケーション102のメイン関数であるアプリケーション関数a(203)がプログラム実行手段108に読み出されると、OS130は、アプリケーション関数a(203)に対応するタスク201を生成すると共に、当該タスクにリソースを割り当てる(図4:S402)。以後上記アプリケーション102は、タスク201として管理される。

【0038】

さらに、上記タスクの生成時に上記OS130は、後述するタスク状態記憶手



段 1 1 0、強制終了情報記憶手段 1 1 4、及びタスク属性情報記憶手段 1 1 7 に当該タスク 2 0 1 に対応するタスクフラグ 1 1 8 を作成し、当該フラグに例えば“0”を格納する（図 4：S 4 0 3）。又、例えばプログラム実行手段は、タスク属性認識手段 1 1 6 に対して、現在タスク 2 0 1 にて実行されている関数がアプリケーション関数である旨を通知する（図 4：S 4 0 4）。当該タスク 2 0 1 にて実行されている関数がアプリケーション関数である旨の通知を受信すると、上記タスク属性認識手段 1 1 6 は、タスク属性情報記憶手段 1 1 7 のタスク 2 0 1 に対応するタスクフラグ 1 1 8 に、例えばアプリケーション関数が実行されていることを示す“1”を格納する。ここで、例えば OS より提供されるアプリケーションによりタスクが作成された場合、通常、メイン関数はシステムライブラリ関数であるため、上記タスク属性情報記憶手段 1 1 7 のタスク 2 0 1 に対応するタスクフラグ 1 1 8 には“0”が格納される。

## 【0 0 3 9】

以後、上記タスク 2 0 1 のアプリケーション関数 a（2 0 3）の先頭のコマンド（命令）より順次実行される。又、必要に応じてアプリケーション関数 b（2 0 4）等の他の関数がプログラム実行手段 1 0 8 に読み出される。尚、上記コマンドが例えば 1 つ実行されるに際して、当該タスク（ここではタスク 2 0 1）に強制終了命令が送信されているか否かをタスク実行判断手段 1 1 1 に問い合わせるが詳細は後述する。

## 【0 0 4 0】

以上が、新規アプリケーションが実行された際に行われる処理（図 4 における初期処理 4 0 0）である。尚、理解に供するために、プログラム実行手段 1 0 8 では、アプリケーション関数 d（2 0 8）及びアプリケーション関数 e（2 0 9）により構成されるタスク 2 0 2 も実行されるものとする。

## 【0 0 4 1】

次に、タスク内において実行されている関数が、強制終了禁止命令を発行した際の、タスク状態設定手段 1 0 9 の処理の詳細について説明する。

## 【0 0 4 2】

例えば、アプリケーション関数 a（2 0 3）の処理中の所定のポイント 2 1 3

にて、タスク201からOS130に対して強制終了禁止命令214が送信されると、当該強制終了禁止命令214はタスク状態設定手段109にて受信される。上記強制終了禁止命令214を受信したタスク状態設定手段109は、タスク状態記憶手段110内の、タスク201に対応するフラグエリア121に例えば“1”を格納する。当該フラグエリア121に格納される“1”は、タスク201が現在強制終了禁止区間にある（強制終了禁止である）ということの意味するものである。

## 【0043】

又、例えばアプリケーション関数a（203）の処理中の所定のポイント215にて、タスク201からOS130に対して強制終了許可命令216が送信されると、当該強制終了許可命令216は上記タスク状態設定手段109にて受信される。上記強制終了許可命令216を受信したタスク状態設定手段109は、タスク状態記憶手段110内の、タスク201に対応するフラグエリア121に例えば“0”を格納する。当該フラグエリア121に格納される“0”は、現在実行されているタスク201が、現在強制終了禁止区間に無い（強制終了禁止ではない）ということの意味するものである。

## 【0044】

以上のように、タスク状態記憶手段110には、各タスクの状態が強制終了禁止（フラグ“1”）であるか、強制終了可能（フラグ“0”）であるかの情報が常時格納される。以後、例えばタスク実行判断手段111よりタスク状態記憶手段110内の各タスクフラグの内容について問い合わせがあると、タスク状態設定手段109は当該タスクフラグに格納されている値を読み出し、上記タスク実行判断手段111に返信する。

## 【0045】

次に、タスク内において実行されている関数が変わる際の、タスク属性認識手段116の処理の詳細について説明する。

## 【0046】

アプリケーション関数a（203）が例えばアプリケーション関数b（204）を呼び出し、当該タスク201における処理がアプリケーション関数b（20

4) に切り替わった場合、上記タスク 201 はタスク属性認識手段 116 に対して、現在タスク 201 にて実行されている関数が、アプリケーション関数である旨を通知する。ここで、上記タスク属性認識手段 116 は、タスク 201 に対応するフラグエリア 119 に “1” を格納する。

## 【0047】

又、アプリケーション関数 b (204) がシステムライブラリ関数 a (205) を呼び出した際にも、タスク 201 は上記タスク属性認識手段 116 に通知を行うが、この場合、タスク 201 にて実行される関数はシステムライブラリ関数であるため、上記フラグエリア 119 には “0” を格納する。尚、上記同様、システムライブラリ関数 a (205) がアプリケーション関数 c (206) を呼び出した際には、上記フラグエリア 119 には “1” が格納される。

## 【0048】

尚、アプリケーション関数 c (206) の処理が終了し、システムライブラリ関数 a (205) に処理が戻った場合にも、上記呼び出した際と同様に、以後実行される関数（処理が戻った後の関数：ここでは “0”）に対応するフラグ “0” 又は “1” がフラグエリア 119 に格納される。

## 【0049】

以上のように、関数の呼び出し時、及び処理が戻った時に、タスクがタスク属性認識手段に以後実行される関数の種類を通知することで、タスク属性情報記憶手段には、現在実行されているタスクの種類が常時格納されることになる。以後、例えばタスク実行判断手段 111 よりタスク属性情報記憶手段 117 内の各タスクフラグの内容について問い合わせがあると、タスク属性認識手段 116 は当該タスクフラグに格納されている値を読み出し、上記タスク実行判断手段 111 に返信する。

## 【0050】

さて、現在上記プログラム実行手段 108 においては、タスク 201 及びタスク 202 が実行されていることは上述したとおりである。ここで、タスク 202 は、上記タスク 201 に対して強制終了命令 211 を発行するタスクであるものとする。上記強制終了命令 211 が発行されるタイミング（ポイント）は、OS

130によるタスクの切り替えのタイミングによって異なるため、各タイミングでの詳細な処理は後述するものとし、ここでは、上記強制終了命令211が発行された際の、強制終了設定手段113の処理の詳細について説明する。

#### 【0051】

まず、所定のポイント（例えば図2に示したポイント207）にて、タスク201からタスク202に処理が切り替わったものと仮定する。当該タスクの切り替えは、従来技術にて説明したものである。

#### 【0052】

ここで、タスク202がタスク201に対して強制終了命令211を発行した場合、具体的には、例えばアプリケーション関数d（208）にて呼び出されたアプリケーション関数e（209）が、所定のポイント210にて発行した強制終了命令211は、強制終了設定手段113にて受信される。当該強制終了設定手段113は、上記強制終了命令211を受信すると、強制終了情報記憶手段114の、タスク201に対応するフラグエリア120に“1”を格納する。該フラグエリア120に格納される“1”は、フラグエリア120に対応するタスク201が強制終了命令を受けた状態であることを意味する。以後、例えばタスク実行判断手段111より強制終了情報記憶手段114内の各タスクフラグの内容について問い合わせがあると、強制終了設定手段113は当該タスクフラグに格納されている値を読み出し、上記タスク実行判断手段111に返信する。

#### 【0053】

以上のように、例えば他のタスクからの強制終了命令は、強制終了設定手段113を介して強制終了情報記憶手段114内の、該当するタスク（終了命令を受けるタスク）のフラグにて示される。

#### 【0054】

以上がタスク状態設定手段109、タスク属性認識手段116、及び強制終了設定手段113における各処理である。

#### 【0055】

次に、上述したタスク状態設定手段109、タスク属性認識手段116、及び強制終了設定手段113の各処理を踏まえた上で、新規アプリケーションが実行

された後に行われる処理の詳細について説明する。

【0056】

図4に示した初期処理400を完了した後、プログラム実行手段108は、上記タスク201のアプリケーション関数a(203)の先頭のコマンド(命令)を順次実行する(図4:S405)。ここで、上記プログラム実行手段108は、コマンドが例えば1つ実行されるに際して、当該タスク(ここではタスク201)に強制終了命令が送信されているか否かを、例えばOS130の機能として提供されているタスク実行判断手段111に問い合わせる。上記タスク実行判断手段111は、上記問い合わせを受け取ると、上記強制終了設定手段113に対して強制終了情報記憶手段114のタスク201フラグ(120)の内容を問い合わせる。尚、当該タスクフラグの内容は上述したとおりである。

【0057】

ここで、タスク201フラグ(120)が“1”ではないと上記強制終了設定手段113より返信があった場合、上記タスク201は強制終了命令を発行されていないことを示し、上記タスク実行判定手段111はプログラム実行手段108に対しその旨を通知することで、当該プログラム実行手段108は次のコマンドを実行する(図4:S406No→S405)。尚、次のコマンドが例えばアプリケーション関数b(203)を呼び出すコマンドである場合等には、タスク属性情報記憶手段117のタスク201フラグ(119)の書き換えを随時行うのは上述したとおりである。

【0058】

又、タスク201フラグ(120)が“1”であると上記強制終了設定手段113より返信があった場合、上記タスク201は強制終了命令を発行されていることを示し、さらに上記タスク実行判断手段111は、タスク状態設定手段109に対してタスク状態記憶手段110のタスク201フラグ(121)の内容を問い合わせる(図4:S406Yes→S407)。

【0059】

ここで、タスク201フラグ(121)が“1”ではないと上記タスク状態設定手段109より返信があった場合、上記タスク201は強制終了禁止区間でな

い、即ち、当該タスク 201 を終了してもシステム全体に影響を及ぼさない（共有リソースに対して操作を行っていない）ことを表している。従って、上記タスク実行判定手段 111 はプログラム実行手段 108 に対しその旨を通知することで、当該プログラム実行手段 108 は、実行している関数（例えばアプリケーション関数 c（206））及び当該関数を呼び出した関数（例えばアプリケーション関数 a（203）、アプリケーション関数 b（204）、システムライブラリ関数 a（205））をすべて終了し、タスク 201 を終了する（図 4：S407 No→S408→End）。

## 【0060】

又、タスク 201 フラグ（121）が“1”であると上記タスク状態設定手段 109 より返信があった場合、上記タスク 201 は強制終了禁止区間であり、即ち、当該タスク 201 を終了するとシステム全体に影響を及ぼす可能性がある（共有リソースに対して操作を行っている等）ことを表している。この場合にはさらに、上記タスク実行判断手段 111 を構成するカレント関数実行判定手段 112 が、タスク属性認識手段 116 に対して、タスク属性情報記憶手段 117 のタスク 201 フラグ（119）の内容を問い合わせる（図 4：S407 Yes→S409）。

## 【0061】

ここで、タスク 201 フラグ（119）が“1”ではない（“0”である）と上記タスク属性認識手段 116 より返信があった場合、上記タスク 201 は現在システムライブラリ関数を実行していることを意味し、即ち、当該システムライブラリ関数を終了するとシステム全体に影響を及ぼす可能性がある（共有リソースに対して操作を行っている等）ため、当該システムライブラリ関数の実行を強制終了させること無く、次のコマンドの処理に移る（図 4：S409 No→S405）。

## 【0062】

又、タスク 201 フラグ（119）が“1”であると上記タスク属性認識手段 116 より返信があった場合、上記タスク 201 は現在アプリケーション関数を実行していることを意味し、当該アプリケーション関数を終了してもシステム全

体に影響を及ぼす可能性が無い（低い）ため、当該実行している関数を終了する（図4：S409Yes→S410）。尚、当該実行している関数がメイン関数であった場合には、強制終了するとシステム全体に影響を及ぼす可能性があるためタスク201を終了する（図4：S410→S405→End）。

## 【0063】

以上のように、本実施の形態においては、強制終了情報記憶手段に格納されている強制終了情報（強制終了命令受信の有無）と、タスク状態記憶手段に格納されているタスク状態（強制終了禁止区間であるか否か）に加えて、タスク属性情報記憶手段に格納されている、現在のタスクにて実行されている関数の属性情報（アプリケーション関数か、システムライブラリ関数か）を参照・判定している。従って、強制終了禁止区間であっても、アプリケーション関数を実行している場合には、強制終了（当該アプリケーション関数の中断）を可能とし、一方でシステムに障害をもたらす可能性のあるシステムライブラリ関数の実行時には、当該タスクを強制終了させないといった強制終了についての柔軟な制御が可能である。

## 【0064】

続いて、図3、図4を用いて、強制終了命令を受けた各ポイントによって異なる処理の詳細を説明する。尚、当該各処理は、上記図4のフローチャートを場合分けして表すものであって、内容が異なるものではない。

## 【0065】

まず、強制終了命令をアプリケーション関数aの強制終了禁止区間設定前に受けた場合（強制終了命令301、309）、アプリケーション関数aを終了し、即ちタスク201を終了する（図4：S407No→S408）。

## 【0066】

強制終了命令をアプリケーション関数a実行時の強制終了禁止区間設定後に受けた場合（強制終了命令302、308）、アプリケーション関数aを終了し、即ちタスク201を終了する（図4：S407Yes→S409→S410→S405→End）。

## 【0067】

強制終了命令をアプリケーション関数 b 実行時に受けた場合（強制終了命令 303、307）、アプリケーション関数 b を終了し、アプリケーション関数 a に処理が戻るが、アプリケーション関数 a も直ちに終了することになり、即ちタスク 201 を終了する（図 4：S407Yes→S409Yes→S410→S405 を 2 回繰り返して End へ）。

## 【0068】

強制終了命令をシステムライブラリ関数 a 実行時に受けた場合（強制終了命令 304、（強制終了命令 306 を含む））、当該システムライブラリ関数 a の実行を終了せずに、アプリケーション関数 c を呼び出す。ここで、当該アプリケーション関数 c は直ちに終了するため（後述する強制終了命令 305 のケース）再度ポイント 320 よりポイント 321 までシステムライブラリ関数 a を実行し、アプリケーション関数 b に処理を戻す。続く当該アプリケーション関数 b 及びアプリケーション関数 a は直ちに終了し、即ちタスク 201 が終了する（図 4：S407Yes→S409No→S405→S406Yes→S407Yes→S409Yes→S410（アプリケーション関数 c 終了）→S405→・・・→S409No→S405・・・（システムライブラリ関数 a 終了後）→S409Yes→S410（アプリケーション関数 b 終了）→S405→・・・→S410（アプリケーション関数 a 終了）→End へ）。

## 【0069】

強制終了命令をアプリケーション関数 c 実行時に受けた場合（強制終了命令 305）、アプリケーション関数 c を終了し、システムライブラリ関数 a のポイント 320 に処理が戻り、ポイント 321 まで処理が実行された後、アプリケーション関数 b、アプリケーション a に処理が戻ると共に直ちに処理が終了し、即ちタスク 201 が終了する（図 4：S407Yes→S409Yes→S410（アプリケーション関数 c 終了）→S405→・・・→S409No→S405→・・・→S409No→S405・・・（システムライブラリ関数 a 終了後）→S409Yes→S410（アプリケーション関数 b 終了）→S405→・・・→S410（アプリケーション関数 a 終了）→End へ）。

## 【0070】



以上が本実施の形態1における処理である。但し、本実施の形態1において、強制終了禁止区間を設定可能である場合の処理について説明したが、必ずしも強制終了禁止区間の判定は必要ではない。即ち、強制終了禁止区間であるか否かの判定をすることなく、現在実行されている関数の属性情報のみに基づいて強制終了の可否を判定してもよい。この場合には、図4に示したS407及びS408の処理を除くのみでよい。

## 【0071】

又、上記関数の属性として、アプリケーション関数とシステムライブラリ関数を例にあげたが、特に上記2つに限定する必要は無く、使用する言語によってさらに異なった関数の場合分けが可能である場合には、当該異なる関数の属性の情報を判断基準として加えてもよい。

## 【0072】

## 〔実施の形態2〕

次に、実施の形態2では、図1を用いてタスク属性記憶手段のタスクフラグに情報を格納する際の処理について説明する。

## 【0073】

上記実施の形態1では、プログラム実行手段108が現在実行している関数の属性（アプリケーション関数、システムライブラリ関数）を、直接タスク属性認識手段116に通知することにより、タスク属性情報記憶手段117の各タスクフラグに当該属性を格納していた。

## 【0074】

即ち、プログラム実行手段108は、現在実行している関数が変更になった場合、当該変更になった関数名（これから実行する関数の名称）を上記タスク属性認識手段116に送信する。当該タスク属性認識手段116は、上記関数名を受信すると、一時記憶手段105に格納されているアプリケーション関数106やシステムライブラリ関数107を参照することで、当該関数がアプリケーション関数かシステムライブラリ関数かを判断する。上記関数の属性を判断したタスク属性認識手段116は、当該関数の種類を該当するタスクフラグに格納する。

## 【0075】

以上により、直接プログラム実行手段より関数属性をタスク属性認識手段116に送信できない場合であっても、タスク属性情報記憶手段は、該当するタスクフラグに現在実行されている関数の属性を格納することが可能になる。

## 【0076】

ところで、上記実施の形態1、及び実施の形態2においては、強制終了を実行する判断の基準に、タスク内で現在実行されている関数の属性を用いている。しかしながら、上記判断の基準を必ずしも関数の属性のみに限定する必要は無い。即ち、マルチタスクOSでは、OSの機能の一部を独立したタスクとして実行することが可能であり、これをシステムタスク（システムプロセス、システムスレッド等）と呼ぶ。又、これに対して、ユーザーが起動したタスクがアプリケーションタスク（ユーザープロセス、アプリケーションスレッド等）である。

## 【0077】

上記システムタスクとアプリケーションタスクは、タスク属性ということが可能であり、当該2つのタスク属性を比較した場合には以下のようなことが言える。

## 【0078】

即ち、上記システムタスクは、OSの機能として提供されているために信頼性（安全性）が高く、一方上記アプリケーションタスクは悪意を持った処理を行う可能性があり、上記システムタスクに比較して信頼性が低い。さらに、上記システムタスクは強制終了されるとシステムに障害をもたらす可能性が高いが、上記アプリケーションタスクは当該タスクがシステムライブラリ関数を利用しているものであったとしても、一般的に強制終了された場合の影響は当該アプリケーションタスク内で完結するものである。

## 【0079】

従って、以上のことを考慮すると、上記判断の基準を関数の属性ではなく、タスクの属性で判断することができるといえる。即ち、上記タスク属性情報記憶手段が記憶するタスクフラグに、関数の属性ではなくタスクの属性を格納するのである。当該格納する処理は、タスクが起動した時の上記初期処理時に、プログラム実行手段がタスク属性認識手段に対し一回だけ行うのみでよい。格納する値は

、例えばシステムタスクの場合には“0”であり、アプリケーションタスクの場合には“1”である。以後、タスク属性情報記憶手段内の該当するタスクフラグのチェックは同様である。

#### 【0080】

以上のように、関数の属性に替えてタスクの属性を管理し、当該タスクに対する強制終了の処理の実行を判断することで、強制終了についての柔軟な制御が可能である。

#### 【0081】

##### 【発明の効果】

以上のように、本発明によれば、関数の呼び出し時、及び処理が戻った時に、タスクがタスク属性認識手段に以後実行される関数の種類を通知することで、タスク属性情報記憶手段には、現在実行されているタスクの種類が常時格納される構成としている。このため、タスク属性情報記憶手段に格納されている、現在のタスクにて実行されている関数の属性情報を参照・判定可能となり、従って、強制終了禁止区間であっても、所定の関数を実行している場合には、強制終了可能とし、一方でシステムに障害をもたらす可能性のある別の関数の実行時には、当該タスクを強制終了させないといった強制終了についての柔軟な制御が可能である。

#### 【0082】

又、タスク属性情報記憶手段が、関数の属性に代えてタスクの属性を格納することにより、タスクの属性に基づいた強制終了の制御が可能である。

##### 【図面の簡単な説明】

#### 【図1】

本発明に係るデジタル機器における概略機能ブロック図。

#### 【図2】

本発明に係るタスク間にわたる強制終了命令発行時のイメージ図。

#### 【図3】

各関数実行時における強制終了命令受信のポイントを示す図。

#### 【図4】

タスク実行時の強制終了命令処理に関するフローチャート。

【図 5】

従来技術におけるプロセス、スレッド及びOSのイメージを示す図。

【図 6】

タスク内の関数構造の一例を示す図。

【図 7】

従来技術におけるタスク間にわたる強制終了命令発行時のイメージ図。

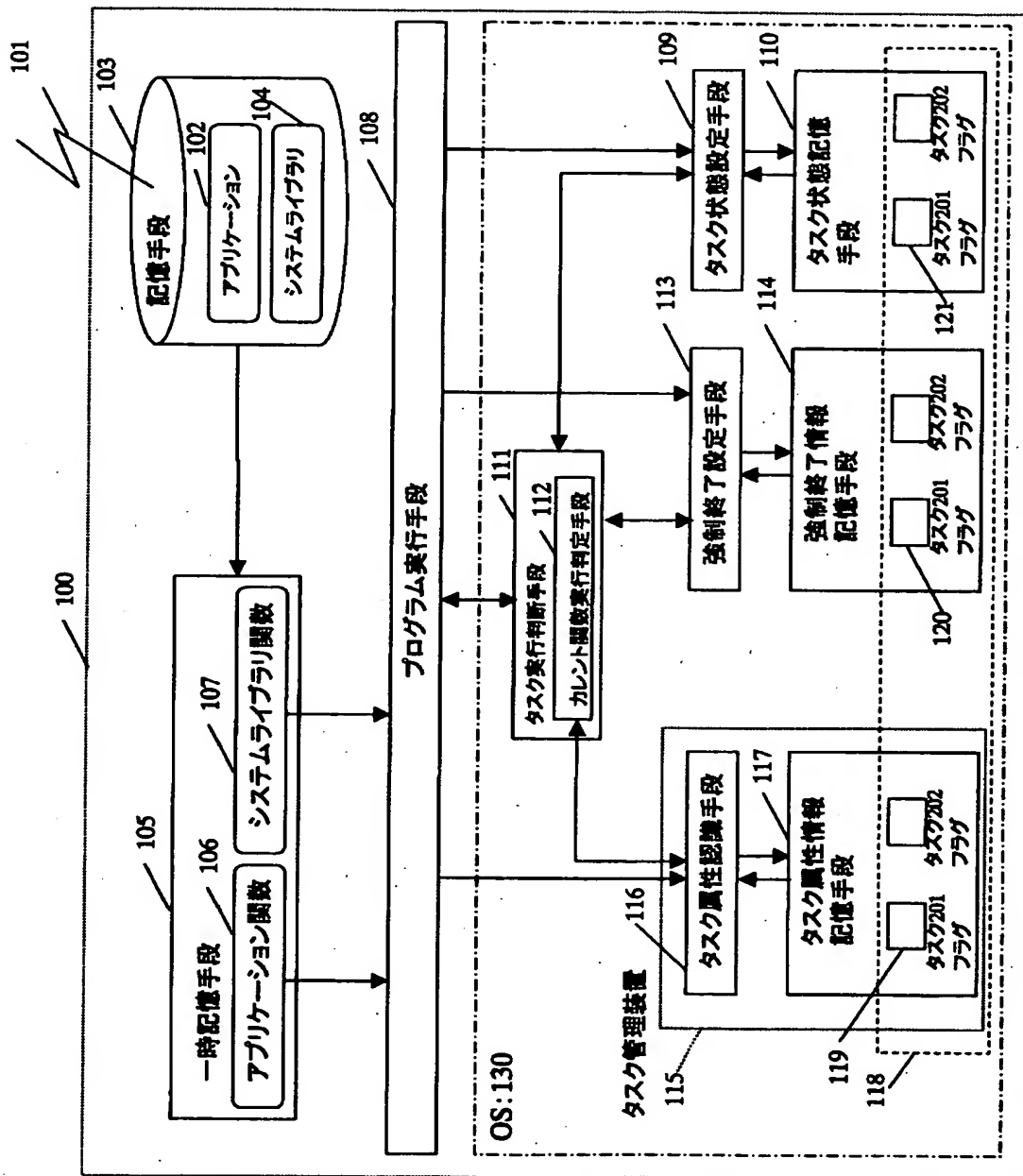
【符号の説明】

- 1 0 0 デジタル機器
- 1 0 1 ネットワーク
- 1 0 2 アプリケーション
- 1 0 3 記憶手段
- 1 0 4 システムライブラリ
- 1 0 5 一時記憶手段
- 1 0 6 アプリケーション関数
- 1 0 7 システムライブラリ関数
- 1 0 8 プログラム実行手段
- 1 0 9 タスク状態設定手段
- 1 1 0 タスク状態記憶手段
- 1 1 1 タスク実行判断手段
- 1 1 2 カレント関数実行判定手段
- 1 1 3 強制終了設定手段
- 1 1 4 強制終了情報記憶手段
- 1 1 6 タスク属性認識手段
- 1 1 7 タスク属性情報記憶手段
- 1 1 8 タスクフラグ
- 1 1 9、1 2 0、1 2 1 フラグエリア
- 1 3 0 OS (OS機能)

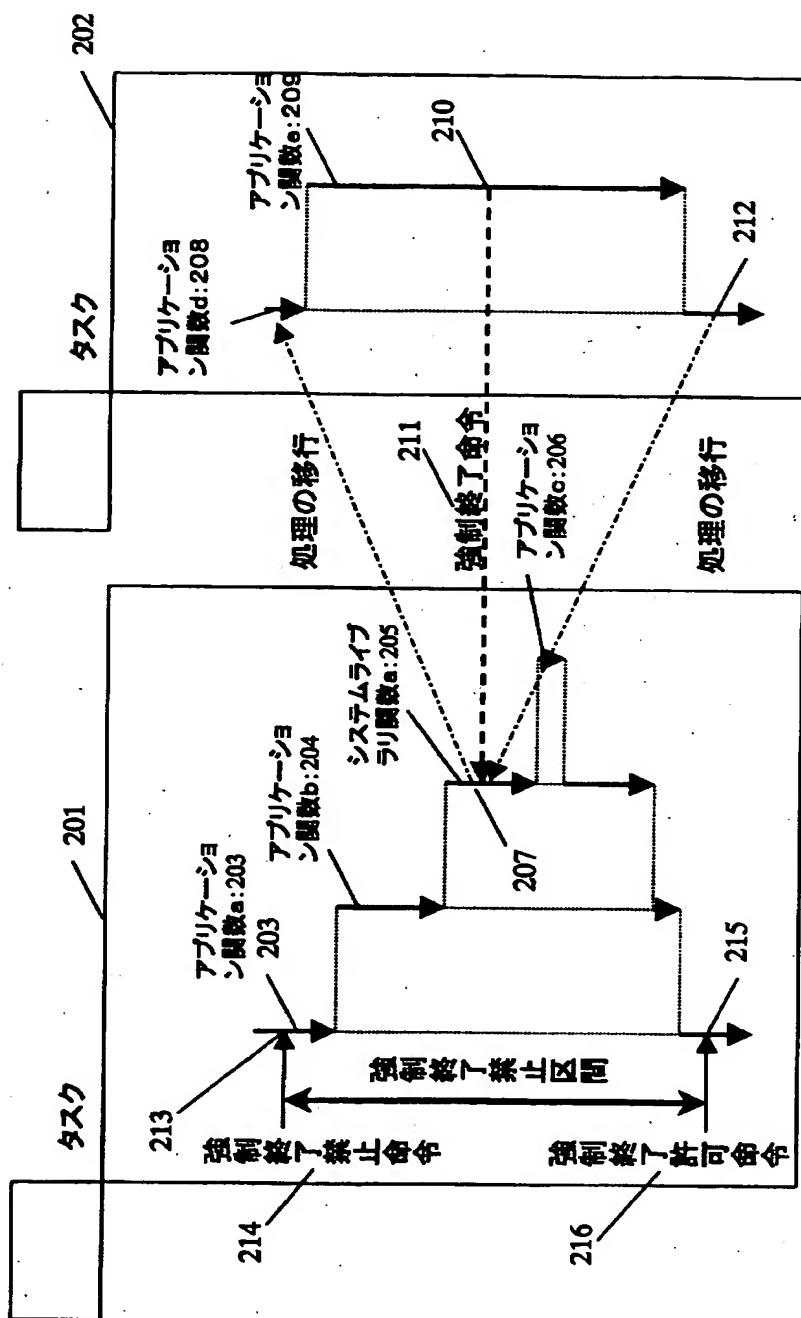
【書類名】

図面

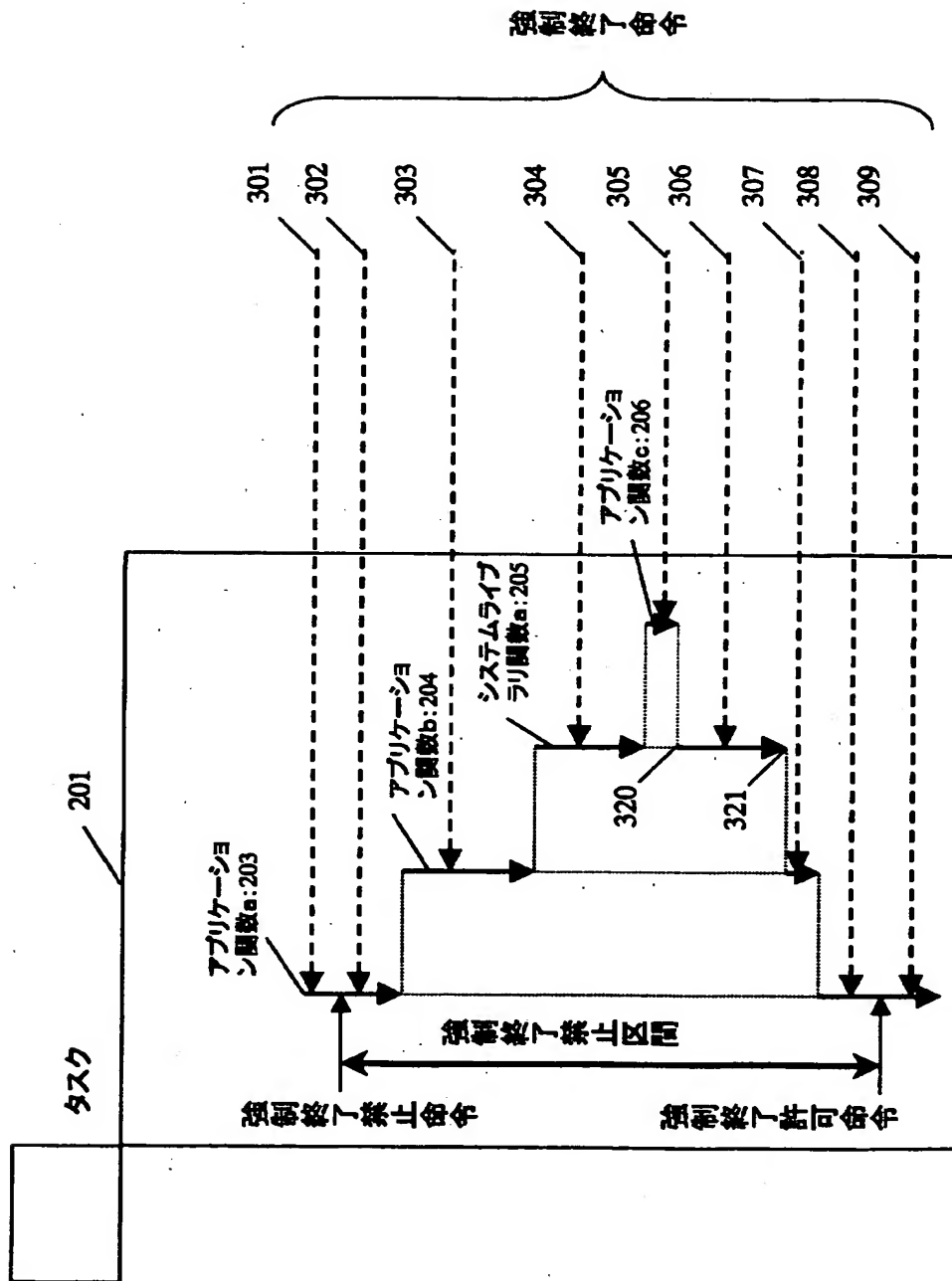
【図 1】



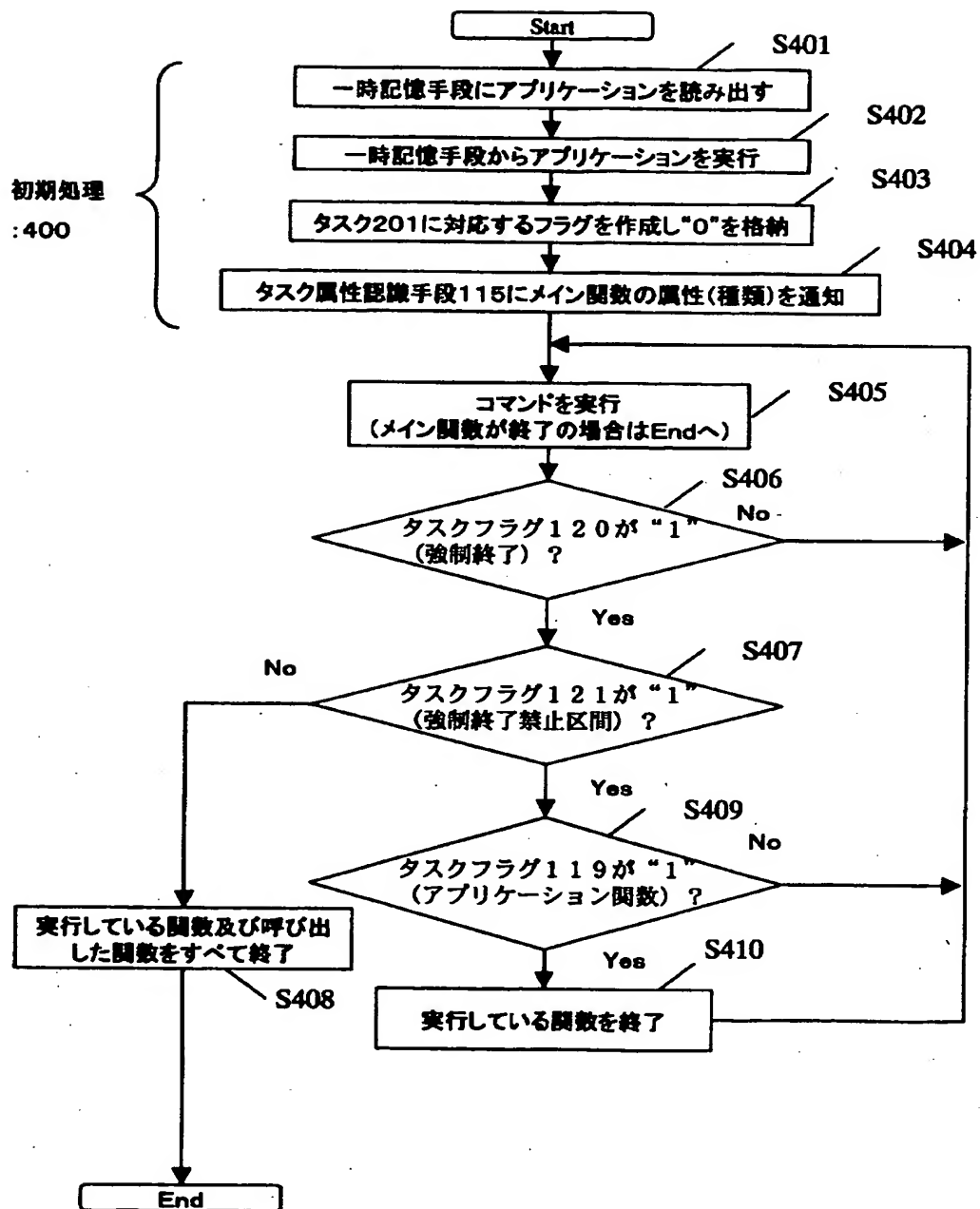
【図2】



【図 3】

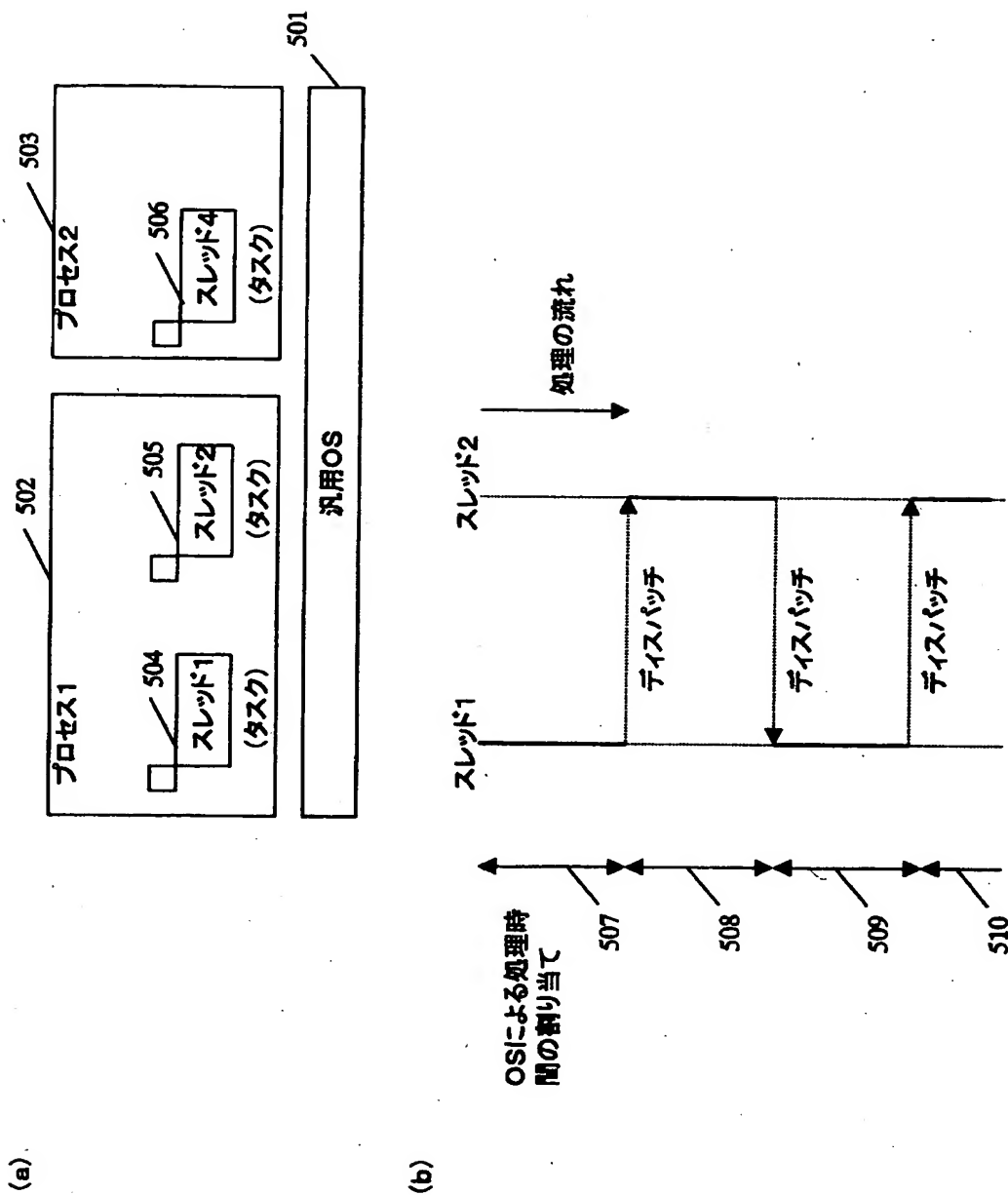


【図 4】

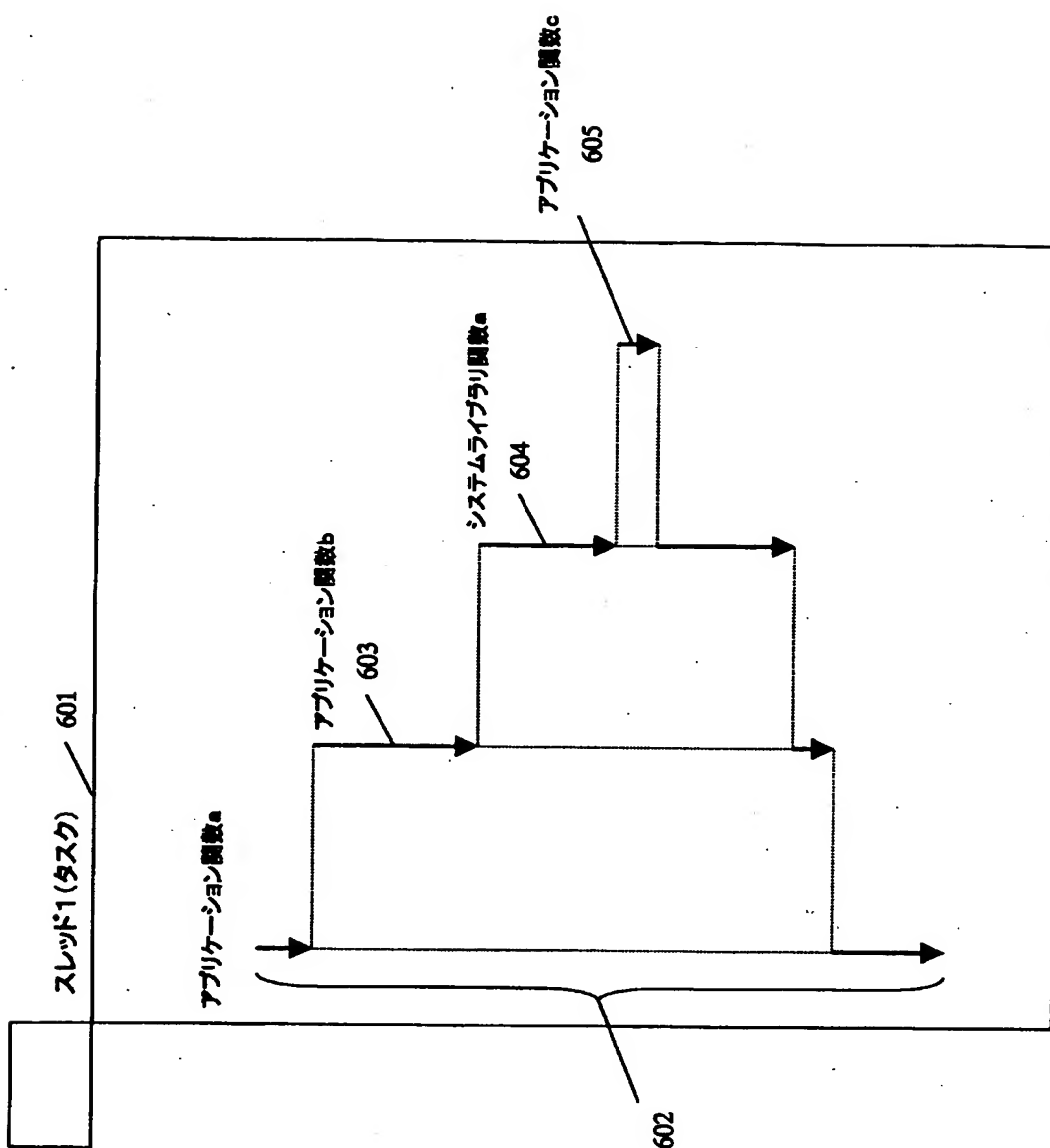




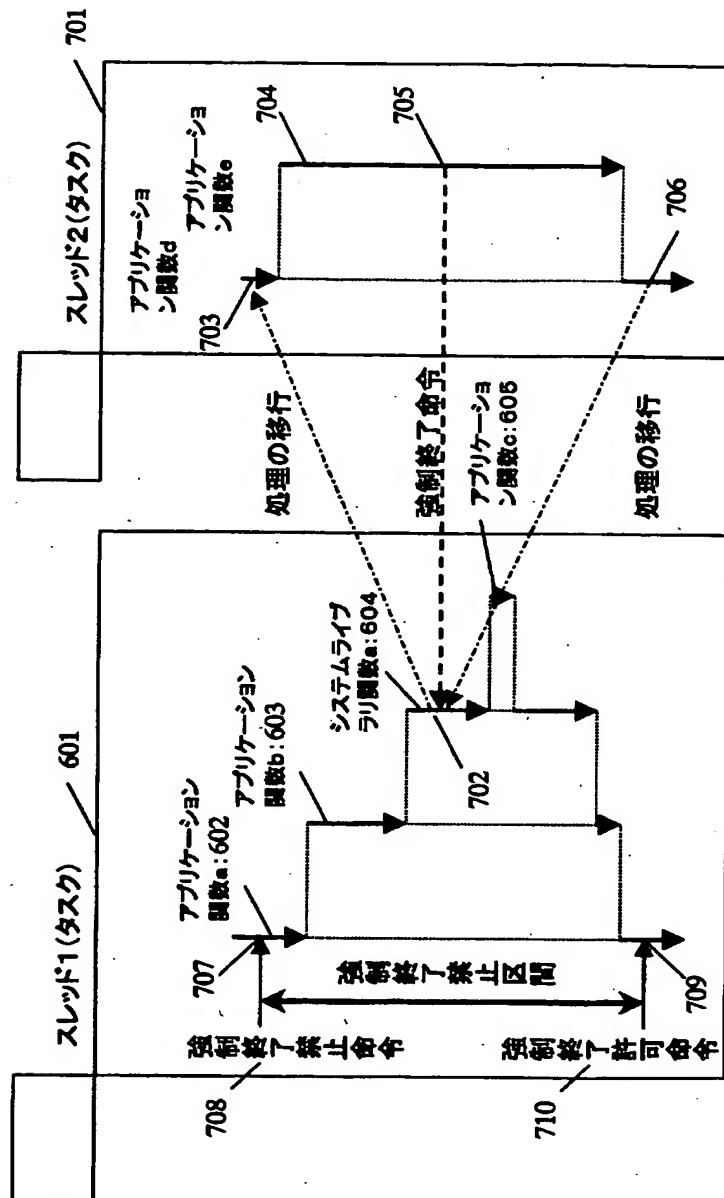
【図 5】



【図 6】



【図 7】



【書類名】 要約書

【要約】

【課題】 システムに障害をもたらす可能性のあるタスクの操作中には、当該タスクを強制終了させないタスク管理方法を提供する。

【解決手段】 プログラム実行手段は、実行されている関数の属性に関する情報を送信すると共に、当該関数の属するタスクに対する強制終了の実行を問い合わせる。又、タスク事項判断手段は、実行されている関数の属性に基づいて上記強制終了の実行を判定する。さらに、タスク属性認識手段は、実行されている関数の属性をタスク属性情報記憶手段に格納し、タスク実行手段からの関数の属性の問い合わせに対して、格納した関数の属性を返信する。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [000005821]

1. 変更年月日	1990年 8月28日
[変更理由]	新規登録
住 所	大阪府門真市大字門真1006番地
氏 名	松下電器産業株式会社